Paradigma Lógico

Aula 06

Profª. M. Sc. Luciana De Nardin

Histórico

- Criado por Alain Colmerauer por volta de 1970 na Universidade de Marselha, França
- ***PRO****gramming in* ***LOG****ic*
- Objetivo inicial → servir de apoio a sistemas de linguagem natural
- PROLOG e LISP (anos 50) linguagens específicas para o desenvolvimento de sistemas de IA

Linguagem Prolog

- Sintaxe e semântica bem diferentes das linguagens procedurais (ou imperativas) e das funcionais
- Pascal:
  - Entrada → programa → saída
- PROLOG:
  - Pergunta → mundo → resposta

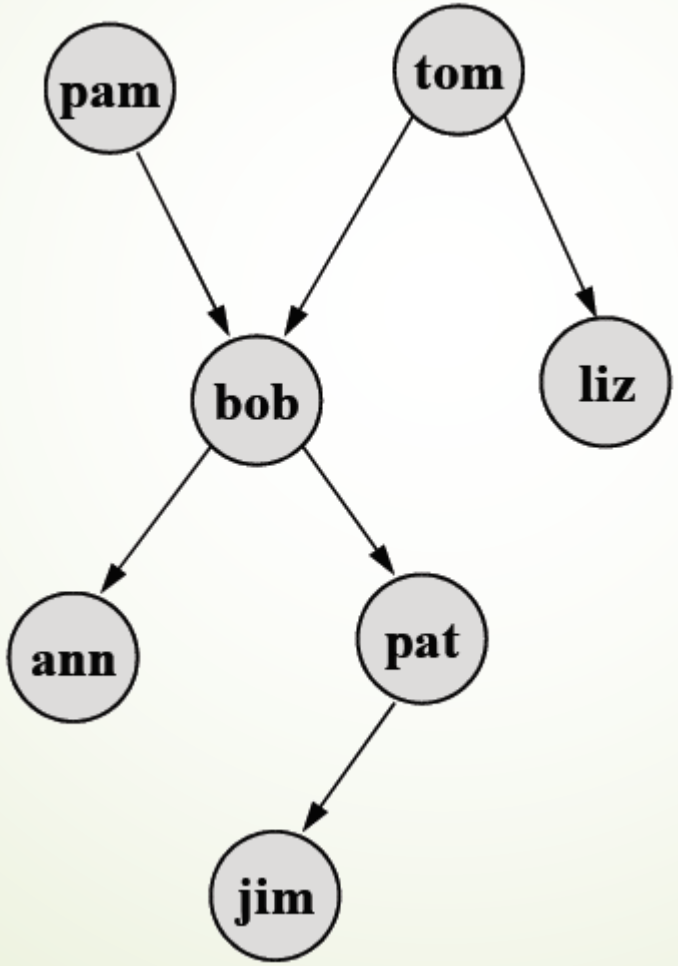
Linguagem Prolog (cont.)

- Consiste em:
  - Declarar fatos ou assertivas;
  - Estabelecer regras ou procedimentos; e,
  - Fazer perguntas
- Os fatos e as regras são fornecidos e a linguagem usa dedução para obter respostas para as questões

Linguagem Prolog (cont.)

- **Fatos**
  - Denotam uma verdade incondicional e estabelece uma relação
  - Sintaxe:
    - Predicado(arg1, arg2, arg3)
      - Argumentos são objetos quaisquer
      - Predicado é a relação que une esses objetos

Linguagem Prolog (cont.)




```
pai(pam,bob).
pai(tom,bob).
pai(tom,liz).
pai(bob,ann).
pai(bob,pat).
pai(pat,jim).
```

Letras maiúsculas para variáveis e minúsculas para átomos

Linguagem Prolog (cont.)

- **Regras**
  - As regras definem condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira
  - Sintaxe:
    - predicado (argumentos) :- predicado (argumentos).

Cabeça ← ..... SE ..... → Corpo

Cabeça é um termo
Cauda é uma conjunção de termos separados por "," (e) ou ";" ou
:- representa um SE

Linguagem Prolog (cont.)

- **Escopo de Variável**
  - Se limita a cláusula onde ela se encontra
  - Exemplo:

```
gosta(pedro,X) :- bonita(X), alegre(X).
gosta(pedro,X).
ama(jorge,X).
```

Linguagem Prolog (cont.)

- Ao receber uma pergunta o PROLOG trata cada um dos termos dela como metas a serem verificadas na base de dados (descrição do mundo)

- Se a meta casa com um fato ela é tida como verdadeira e diz-se que a meta foi satisfeita

Linguagem Prolog (cont.)

- Se a meta casa com a cabeça da regra, ela produz submetas para cada termo da cauda da regra. Se uma meta não casa com nenhum fato nem com a cabeça de nenhuma regra, ela falha, ou seja, é considerada falsa.
- A falha de uma meta desencadeia o backtracking

```
gosta(pedro,X) :- bonita(X), alegre(X).
```

Linguagem Prolog (cont.)

- *Backtracking*
  - Estrutura nativa do PROLOG que permite o retrocesso na meta para reconsideração de uma submeta provada anteriormente
  - Para forçar o retrocesso → ";"

Linguagem Prolog (cont.)

- Prolog suporta variáveis de números inteiros e seus cálculos
- Atribuição é representada por is

  Exemplo:

  A = B / 17 + C ≅ A is B / 17 + C

  B e C devem estar instanciados, senão, erro!!!

- Soma is Soma + 20 é possível?

Linguagem Prolog (cont.)

```
velocidade(ford,100).
velocidade(chevy,105).
velocidade(dodge,95).
velocidade(volvo,80).
tempo(ford,20).
tempo(chevy,21).
tempo(dodge,24).
tempo(volvo,24).

distancia(X,Y) :- velocidade(X,V),
                  tempo(X,T),
                  Y is V * T.
```

Linguagem Prolog (cont.)

- ***Underline***
  - Usa-se o *underline* para indicar irrelevância de um objeto
  - Exemplo:

```
aniversario(maria,data(25,janeiro,1979)).
aniversario(joao,data(5,janeiro,1956)).
signo(Pessoa,aquario):-
                aniversario(Pessoa,data(Dia,janeiro,_)),
                Dia >= 20.
```

Linguagem Prolog (cont.)

- **Conjunção**
  - O operador de conjunção e (,) implica na necessidade de aceitação de todas as condições
  - Exemplo:

```
avos(X,Z) :- pais(X,Y), pais(Y,Z).
```

Linguagem Prolog (cont.)

- **Disjunção**
  - O operador de disjunção ou (;) permite a aceitação de uma ou outra condição
  - Exemplo:

```
amiga(X) :- (X = maria; X = joana).
```

Linguagem Prolog (cont.)

- **Exercícios**
  - 1) Defina uma regra que seja capaz de identificar se um número é positivo ou negativo.
  - 2) Defina uma regra que seja capaz de determinar de um número é par ou ímpar.

Linguagem Prolog (cont.)

- Resolução

```
numero(X) :- X>=0, write('positivo‘).
numero(X) :- X<0, write('negativo').

par_impar(X):- X mod 2 =:= 0, write('par').
par_impar(X):- X mod 2 \= 0, write('impar').
```

Linguagem Prolog (cont.)

- **Controle de Corte (*cut*)**
  - É utilizado quando uma determinada regra é a última a ser analisada e haveria problemas na continuidade das demais regras
  - Representado por **!**

Linguagem Prolog (cont.)

- Exemplo:

```
amigo(joana,ana).
amigo(maria,ana).
amigo(pedro,jose).
amigo(pedro,ana).

um_unico_amigo(X,Y):-amigo(X,Y),!.
```

Linguagem Prolog (cont.)

- **Negação (*not*)**
  - Operador *not* define uma forma particular de negação denominada negação por falha.
  - Exemplo:

```
estudante(jorge).
estudante(claudio).
casado(jorge).
estudante_solteiro(X):- estudante(X),
                        not casado(X).
```

Linguagem Prolog (cont.)

- **Problema do Fatorial**
  - Como o procedimento é recursivo é fundamental encontrar a definição de parada da recursividade
  - Como n não tem limite superior, o cálculo do fatorial deve ser genérico para qualquer n, então, deve-se começar pelos fatos que são conhecidos

Linguagem Prolog (cont.)

- Resolução:

```
fatorial(0,1):-!.
fatorial(N,F) :- N1 is N - 1,
                 fatorial(N1,F1),
                 F is F1 * N.
```